Trabajo de fin de grado

Grado en ingeniería en tecnologías industriales

Diseño y fabricación de un sistema de control de inventario conectado a la red

MEMORIA

Autor: Àlex Martín Pérez
Director: Oriol Boix Aragonès
Convocatoria: Julio 2018

Escola Tècnica Superior
d'Enginyeria Industrial de Barcelona

Resumen

Este proyecto consiste en el diseño y la construcción de un prototipo de un sistema de control de inventario.

En un almacén, los distintos productos están distribuidos en cajas. Unos sensores colocados en la parte inferior de dichas cajas miden el peso de estos productos, y esos datos se envían a la red. Posteriormente, un usuario puede acceder a las mediciones ya sea mediante un teléfono móvil o utilizando un ordenador. Adicionalmente, se pueden mostrar gráficos del comportamiento temporal de estos datos.

Para ello, se ha programado un microcontrolador con conexión a internet mediante Wi-Fi. La programación se ha realizado utilizando el entorno de Arduino, que emplea un lenguaje basado en C++. Se han utilizado células de carga con sus correspondientes amplificadores para pesar los productos. En cuanto al envío de datos a la nube, se ha seleccionado una plataforma para su almacenamiento, gestión y visualización. La escogida ha sido Adafruit IO, que cuenta con abundante documentación y además está adaptada para su uso con dispositivos móviles. Se ha desarrollado un programa para que realice las mediciones y envíe estos datos a la plataforma previamente mencionada. Adicionalmente, para calibrar las células de carga se ha desarrollado otro programa que hace uso de la conexión serie de la placa con un ordenador.

Para proteger los sensores de posibles sobrecargas se ha incorporado un zumbador que emite sonidos cuando el peso de alguna de las básculas llega a un valor concreto cercano al límite. Otra funcionalidad añadida consiste en el envío de mensajes al teléfono móvil si algún sensor detecta un peso demasiado bajo en un producto. De esta forma, se puede realizar un nuevo pedido y no quedarse sin inventario.

Finalmente, se ha realizado el montaje del prototipo y se han propuesto algunas posibles mejoras, como podría ser el uso de sensores integrados en la estantería o la utilización de un concentrador y un ordenador central para la gestión de las básculas. Otra posible mejora pasaría por unir ambos programas en uno solo e incorporar pequeñas pantallas y teclados en la placa para hacer de la calibración un proceso mucho más rápido.

Índice

1. Introducción

1.1. Objetivos

El objetivo principal de este proyecto es crear un prototipo de un sistema de control de inventario conectado a la red, de forma que se puedan visualizar los datos desde un dispositivo móvil.

En este proyecto se busca integrar conocimientos de diferentes áreas adquiridos a lo largo de esta carrera, combinando así temáticas como la electrónica, la programación o la resistencia de materiales. Por otra parte, el autor pretende introducirse en el ecosistema del internet de las cosas, pues en los últimos años ha habido un auge entorno a la conexión de objetos cotidianos a la red y se prevé que en el futuro sea algo habitual.

1.2. Alcance del proyecto

En este proyecto se pretende construir un prototipo, desde la selección de componentes hasta la fabricación del sistema, pasando por la programación de la placa. Dado que se trata de un prototipo, se ha limitado el número de básculas a dos, pero en una versión final se puede ampliar su número sin realizar apenas cambios. También se contemplan otras opciones de mejora o cambios que se podrían realizar en una versión comercial.

1.3. Motivación

La principal motivación a la hora de realizar este proyecto ha sido la voluntad de mejorar en programación y de aprender otros lenguajes además del que se enseña en la Escuela. También se ha valorado el hecho de trabajar con Arduino, una herramienta con la que se pueden realizar multitud de proyectos diferentes, y de introducirse en el mundo del internet de las cosas.

2. El control de inventario

La logística acostumbra a ser uno de los principales quebraderos de cabeza de cualquier empresa. En concreto, el control de inventario suele basarse en contabilizar las unidades que entran y salen del almacén. Sin embargo, este sistema presenta ciertas desventajas.

Por una parte, se tiene que llevar un control muy preciso sobre todas las unidades. Debido a esto, los desajustes no son infrecuentes. Una posible solución consiste en etiquetar cada producto con un código, de forma que siempre se sepa dónde se encuentra (o debería encontrarse) este. A estos productos se les añaden códigos de barras o etiquetas de radiofrecuencia (RFID) que se escanean en cada sección del proceso. Sin embargo, no es aconsejable etiquetar todos los productos. Por ejemplo, codificar todas las tuercas en un almacén sería demasiado caro en comparación con su coste unitario.

Otra desventaja de este sistema radica en la posible inviabilidad de apuntar todas las entradas o salidas de productos. Por poner otro ejemplo, en un restaurante no sería recomendable que un cocinero, al coger una patata, se dedicara a pesarla e introducir ese valor en el ordenador, pues perdería demasiado tiempo.

Por estas razones, en este proyecto se ha propuesto otra forma de controlar el inventario.

3. Propuesta de funcionamiento del sistema

Para solventar algunos de los problemas planteados en el apartado anterior, se ha propuesto un sistema que hace uso de básculas para conocer con exactitud la cantidad de producto del que se dispone en un determinado momento.

Para ello, cada producto debe colocarse en cajas sujetas por células de carga a la base de la estantería. Estos sensores son los encargados de medir el peso de los productos, y son leídos desde una placa que, a su vez, los envía a la red.

Para una mejor y más cómoda monitorización se pretende que estos datos sean accesibles no solo desde un ordenador, sino también desde dispositivos móviles.

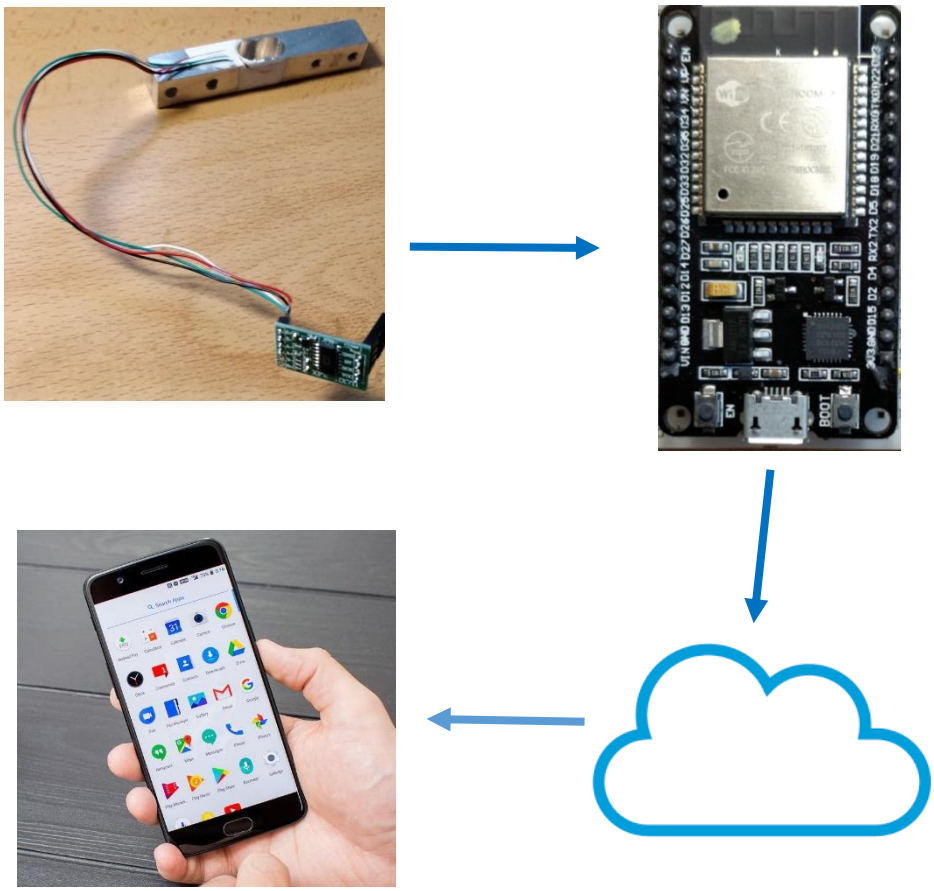

Figura 1. Esquema del funcionamiento del sistema[1]

[1] Las imágenes superiores son del autor del proyecto. Fuentes de las imágenes inferiores:
https://www.enriquedans.com/2018/04/sobre-nubes-y-pymes.html
https://www.theverge.com/2017/6/29/15894850/oneplus-5-screen-jelly-scrolling-statement

4. Selección de componentes y tecnologías

4.1. Elección del tipo de placa

En estos proyectos en los que no es necesario el procesamiento de grandes cantidades de datos y más bien orientados a soluciones domésticas, predominan dos familias de placas: Arduino y Raspberry Pi.

Para ejemplificar mejor sus diferencias, se ha escogido el principal modelo de cada marca. En el caso de Arduino, su placa más popular es el Arduino UNO R3, mientras que por la parte de Raspberry destaca la Raspberry Pi 3 Model B.

La placa Arduino (1) contiene un microcontrolador de ocho bits que opera a una frecuencia de 16 MHz, además de contar con 32 kB de memoria flash (la del programa) y 2 kB de memoria RAM (utilizada para los datos). También tiene una gran cantidad de patas tanto analógicas como digitales, por las cuales puede soportar una corriente de hasta 20 mA. Incorpora también una conexión USB (tipo B) y una entrada para conectarlo a una fuente de alimentación. Cabe remarcar que existen también otras placas, ya sea con mayor potencia de procesamiento, más pequeñas, con más memoria o con un mayor número de patas.

Además, dentro del ámbito de Arduino existen los llamados módulos (*shields* en inglés). Estos módulos son piezas de hardware que se adjuntan a la placa principal para proporcionar ciertas funcionalidades como por ejemplo lectura de tarjetas SD, conectividad a internet o Bluetooth.

Al igual que en el caso del hardware, el software para programar Arduino también es sencillo. Este controlador no dispone de sistema operativo y el único código que ejecuta es el que se escribe en el entorno de programación (IDE) de Arduino. El lenguaje utilizado para programarlo está basado en C++, uno de los más utilizados, cosa que facilita su uso por personas no especializadas en programación.

Por otra parte, tenemos la Raspberry Pi (2). Esta placa es, a diferencia de la Arduino, un pequeño ordenador. Contiene un microprocesador de 64 bits y cuatro núcleos que trabaja a una frecuencia de 1,2 GHz, y una memoria RAM de 1 GB. Al igual que la placa Arduino, tiene una generosa cantidad de patas, pero su potencia está más limitada (10 mA a 3,3 V). En cuanto a conexiones con el exterior, la Raspberry Pi dispone de Wi-Fi, Bluetooth, cuatro puertos USB, un conector micro USB para la alimentación, HDMI, Ethernet, salida de audio y ranura para tarjeta micro SD; todas ellas incorporadas en la placa sin necesidad de adjuntar módulos.

En cuanto a su programación, al ser un pequeño ordenador, incorpora un sistema operativo que suele estar basado en Linux. Por ello, los programas se ejecutan dentro de este entorno tal y como lo harían en nuestro ordenador.

En la Tabla 1 se comparan las principales especificaciones de ambas placas.

	Arduino UNO R3	Raspberry Pi 3 Model B
CPU	8 bits, 1 núcleo, 16 MHz	32 bits, 4 núcleos, 1,2 GHz
Número de patas GPIO	20	26
Máxima corriente en cada pata	20 mA	16 mA
Memoria flash	32 kB	Capacidad de la micro SD insertada
Memoria RAM	2 kB	1 GB
Otras funcionalidades	Puerto USB tipo B, conector de alimentación	Wi-Fi, Bluetooth, cuatro puertos USB, un conector micro USB para la alimentación, HDMI, Ethernet, salida de audio, ranura para Micro SD
Precio oficial	25 €	37 €

Tabla 1. Comparación de las principales especificaciones técnicas de ambas placas

A simple vista podría parecer que la placa Raspberry Pi es claramente mejor a la de Arduino, pero todo depende del proyecto en el que se utilice. Hay que destacar que la placa de Raspberry tiene un consumo mucho mayor. Además, existen placas compatibles con Arduino desde 4 €, mientras que es difícil encontrar placas similares a la Raspberry Pi por debajo de los 20 €. A eso hay que sumarle la gran variedad de placas y módulos programables con el entorno de Arduino que añaden o mejoran ciertas características respecto a la placa básica, con lo que responden más específicamente a las necesidades de cada proyecto.

Otra ventaja de las placas Arduino es su facilidad de programación. A esto cabe sumarle que esta tan solo ejecuta el programa que tiene guardado, a diferencia de las placas Raspberry. Estas últimas, al tener un sistema operativo, no garantizan que el programa se ejecute con máxima prioridad (ni siquiera que esté funcionando) ya que existen múltiples procesos en ejecución. Por todo ello, finalmente se ha decidido escoger una placa que trabaje con el entorno de Arduino.

4.2. Selección del sensor

Para evaluar el peso destacan en el mercado dos tipos de sensores: los sensores de fuerza resistivos y las células de carga.

Los sensores de fuerza resistivos son pequeños sensores que miden la presión que se ejerce sobre su superficie. Su funcionamiento se basa en una reducción de su resistencia interna cuando se aplica una presión.

Sus principales ventajas son sus reducidas dimensiones, su sencilla instalación y la facilidad a la hora de realizar las lecturas. Sin embargo, su precisión no es demasiado elevada (tienen un error de hasta el 25 %) (3), y el propio fabricante recomienda no usarlos si se requieren mediciones precisas. Además, la relación entre su resistencia y el peso aplicado no es lineal, sino logarítmica. Utilizando la conductancia en lugar de la resistencia se podría conseguir una respuesta aproximadamente lineal en ciertos tramos, tal y como se muestra en la Figura 2, pero eso aumentaría todavía más el error. Otra limitación es el peso máximo que soportan, que no puede superar los 100 N. Su precio oscila entre los 5 y los 8 €.

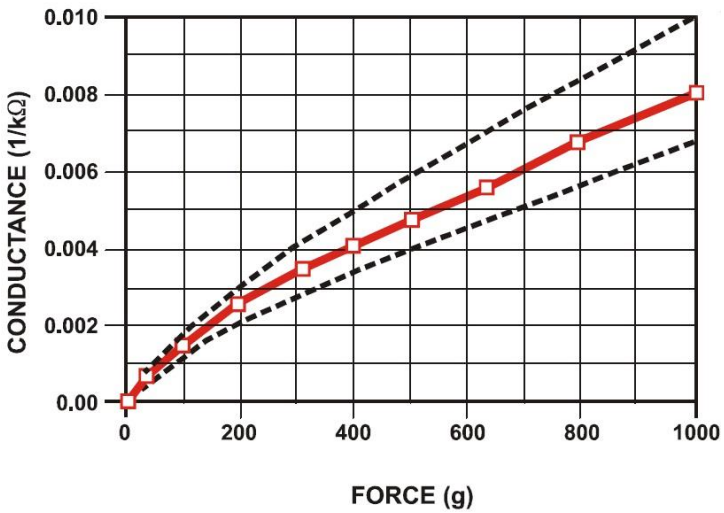

Figura 2. Conductancia en función de la fuerza aplicada (3)

Las células de carga, en cambio, se basan en el uso de galgas extensométricas. Estas galgas hacen uso del efecto piezoresistivo, por el que cambian su resistencia al sufrir una deformación. Las galgas están incorporadas en una barra (normalmente hecha de aluminio o acero) que sujeta el objeto a pesar.

Este sensor tiene la ventaja de contar con un error mínimo de hasta un 0,05 % (4). También puede soportar una masa de hasta 800 kg o incluso superior. Como contrapunto, cabe destacar que el cambio en su resistencia es muy pequeño, por lo que necesita un amplificador para que el microcontrolador pueda hacer una lectura correcta del peso.

Afortunadamente, existe un módulo fabricado con este propósito para el tipo de célula escogida, el HX711, que incluye una librería diseñada para su uso con Arduino. También necesita un mayor espacio de instalación al ser más grande que el primer sensor y el montaje eléctrico es más complejo, ya que necesita más elementos y cables. Además, su programación es más compleja, ya que precisa de una calibración previa y del uso de su propia librería y funciones. El precio del conjunto de célula y módulo parte de los 3 €.

Para una correcta lectura del peso, ambos sensores necesitan un soporte mecánico. En el caso del primer sensor, suponiendo que el objeto a pesar repose sobre una superficie plana (como podría ser una caja), se requiere la instalación de tacos entre el sensor y la superficie, pues la lectura se realiza sobre un área relativamente pequeña. Adicionalmente, haría falta el uso de tres sensores como mínimo para garantizar la estabilidad de la superficie. El peso sería entonces la suma de los datos de los tres sensores. En el caso de la célula de carga, esta incorpora una o dos ranuras (dependiendo del modelo) para poder atornillar a ellas la superficie (o caja) donde se depositará el objeto a pesar. También hay que tener en cuenta la altura de este sensor (que no es menospreciable) a la hora de dimensionar las estanterías que albergarán el producto.

Puesto que en este proyecto se diseña un sistema de control de existencias general, sin especificar si este está destinado a productos con un peso en incrementos constantes (como podrían ser paquetes de azúcar) o con un peso indeterminado (kg de tomate o de patatas), se ha decidido escoger el sensor que más opciones abarca, es decir, el sensor con más precisión. Además, respecto al primer sensor, aunque tuviéramos productos con un peso determinado e igual, bastarían cuatro unidades de ese producto para que la lectura pudiera ser errónea (recordemos que tiene un error de hasta el 25 %, al que hay que sumar la no linealidad de la respuesta). También hay que tener en cuenta que el precio de la célula de carga y el amplificador es bastante inferior al de los sensores de presión al necesitar tres de estos como mínimo. Así pues, pese a necesitar un mayor espacio para su instalación y a pesar de una programación e instalación más complejas, se ha decidido optar por utilizar células de carga.

4.3. Comunicación entre dispositivos

En este proyecto es necesario que, además de la placa (o las placas), un segundo dispositivo (en concreto un teléfono móvil) pueda disponer de los datos de los sensores. Para ello se tiene que establecer un sistema de comunicación entre la placa y el teléfono. En cuanto a la placa, gracias a la gran diversidad de módulos existentes puede disponer de conexiones como Bluetooth, Wi-Fi, Ethernet o Zigbee, entre otras. Por otra parte, el receptor

de esos datos será un teléfono móvil, con conexiones a redes de datos (LTE, HSPA…), Wi-Fi, Bluetooth o NFC. Por tanto, algunas posibles conexiones entre la placa y el móvil podrían darse mediante Bluetooth, conexión a una red local o bien mediante el envío de los datos a un servidor externo en internet por parte de la placa y la posterior obtención de esas mediciones mediante el teléfono móvil.

Bluetooth es seguramente una de las conexiones más populares a corta distancia, ya que permite conectar de forma sencilla y rápida diferentes dispositivos, además de tener un reducido consumo de energía. Sin embargo, la velocidad de transmisión es relativamente baja (pese a que últimamente se está mejorando en este aspecto con las nuevas versiones) y su rango es bastante limitado, llegando a un máximo de 10 m en espacios abiertos (5). Además, para una buena conexión no debe haber obstáculos entre ambos dispositivos.

Una segunda opción es el envío de los datos a una red Ethernet. Esta opción tiene como principal ventaja una mayor velocidad de transferencia de datos. Sin embargo, tanto el Wi-Fi como la conexión por cable Ethernet tienen un mayor consumo de energía.

Esta red puede ser local o conectada a internet. En el caso de una red local, tanto la placa como el teléfono deberían estar conectados a dicha red. Si bien el móvil solo se puede conectar mediante Wi-Fi, la placa también puede hacerlo utilizando un cable Ethernet. Tiene la ventaja de tener un rango mucho mayor que en el caso del Bluetooth, ya que los cables de Ethernet pueden ser relativamente largos y el Wi-Fi alcanza los 200 m, además de que puede haber diferentes puntos de acceso para la red. Otra ventaja de utilizar una red Ethernet local es la seguridad que ofrece esta respecto a la red conectada a internet.

En el caso de que la red Ethernet esté conectada a internet, existe la opción de enviar los datos a un servidor externo (en el apartado 4.6 se explican las diferentes opciones disponibles) para que luego puedan ser obtenidos por el teléfono. Tiene como principal ventaja un rango ilimitado.

Para escoger qué sistema utilizar hay que tener varias cosas en cuenta. En este proyecto, la velocidad de conexión no es importante pues la cantidad de datos a enviar es mínima y perfectamente asequible para cualquiera de los sistemas comentados. En cuanto al consumo energético, Bluetooth es el claro vencedor, pero tiene el inconveniente de tener un rango muy limitado. Por otra parte, el rango de conexión a una red Ethernet local, pese a ser mucho mayor al del Bluetooth, obliga a consultar los datos en el mismo edificio (o en la misma zona) en el que está la placa, mientras que la conexión de dicha red a internet permite una consulta de estos en cualquier parte. Cabe mencionar que existen mecanismos para acceder a una red local desde cualquier lugar mediante una VPN (red privada virtual).

El sistema escogido finalmente ha sido el envío de las mediciones a un servidor externo. El

rango es sin duda un factor decisivo en este proyecto, pues el hecho de poder consultar los datos en cualquier lugar es una funcionalidad determinante (pese a que la conexión a la red tiene un mayor consumo energético). La privacidad de los datos, por otra parte, no es un factor crítico en este proyecto y, por su mayor simplicidad, se ha escogido la opción de la red conectada a internet.

4.4. Conexión de la placa a internet

Para conectar la placa Arduino a internet existen diferentes posibilidades. Las conexiones más comunes se dan mediante Wi-Fi y Ethernet, aunque hay otras opciones, como la conexión a redes de datos (GSM). Esta última es útil únicamente cuando no es posible disponer de un punto de acceso a la red cercano a la placa. En este proyecto, las básculas estarían situadas dentro de un edificio y, por tanto, ubicar un punto de acceso dentro de él no sería un problema. Por ello se estudiarán únicamente las opciones de Wi-Fi y Ethernet (6).

La conexión por Wi-Fi es una conexión inalámbrica con un rango que puede llegar a los 200 m en espacios abiertos, aunque se reduce ante la presencia de obstáculos. Tiene como principal ventaja una mayor simplicidad de montaje al no utilizar cables. Como contrapartida, pueden producirse interferencias con otros dispositivos con comunicación inalámbrica. Además, es una conexión sensiblemente más lenta que el Ethernet, ya que se calcula que solo aprovecha en torno a un 35 % de la velocidad del punto de acceso. A eso hay que sumarle un mayor consumo energético.

La conexión mediante cable Ethernet, a diferencia del Wi-Fi, tiene un rango limitado por la longitud del cable. Tiene la ventaja de proporcionar una conexión más robusta y veloz, pero al hacer uso de cables precisa de una instalación más compleja. En este proyecto, en el que se podrían llegar a utilizar varias placas, se necesitaría llevar estos cables hasta el punto de acceso, con la consecuente utilización de espacio requerida.

Para proporcionar a la placa conexión por Wi-Fi o Ethernet, en general es necesario añadirle un módulo adicional. En el caso del Wi-Fi, estos dispositivos suelen ser más pequeños y baratos que los que proporcionan un puerto Ethernet.

Para decidir qué conexión utilizar, hay que considerar diversos factores. En este proyecto no es necesaria una alta velocidad de transferencia de datos. A esto hay que sumarle que, a medida que se añaden placas, la instalación con cable se va haciendo más voluminosa y los cables pueden alcanzar grandes longitudes. En el caso del uso de Wi-Fi, se pueden conectar múltiples dispositivos al punto de acceso de forma sencilla y eso puede llegar a ser

importante en grandes instalaciones. Además, los módulos son más baratos y por estas razones la conexión por Wi-Fi ha sido finalmente la opción escogida.

4.5. Elección del modelo de la placa

Dentro del ecosistema de Arduino existen muchas placas diferentes, cada una de ellas enfocada a un uso particular. Por ello, es importante escoger correctamente cuál de ellas utilizar.

Para empezar, hay que tener claro qué se busca en la placa. Una de las características importantes es disponer de memoria suficiente, ya que el programa puede llegar a ser complejo e incluye varias librerías que pueden ocupar un espacio considerable. Otro factor a tener en cuenta es que disponga de una cantidad de patas considerable, ya que ello hará que se puedan conectar más básculas a una sola placa. Otro punto que hay que considerar es la posibilidad de que esta incorpore conexión Wi-Fi sin necesidad de añadir un módulo externo (esta característica no es indispensable pero sí deseable). Por último, un factor importante es siempre el precio.

Primeramente, se ha realizado una investigación previa para determinar posibles placas que cumplan estas características. Las escogidas han sido la Arduino MKR1000, la Arduino Mega (junto al módulo ESP8266) y la ESP32 *development board*.

La Arduino MKR1000 (7) es una placa que ha sido diseñada específicamente para proyectos de internet de las cosas, es decir, proyectos que requieren una conexión a la red. Para ello, dispone de conexión Wi-Fi integrada. Tiene una memoria interna de 256 kB y un total de 15 patas de entrada de datos (entre analógicas y digitales). Sin embargo, tiene un precio elevado, en torno a 45 €, y es difícil encontrarla más barata.

La placa Arduino Mega (8) destaca por tener un gran número de patas; en concreto, tiene 56 patas para señales digitales y otras 16 tanto para analógicas como para digitales, sumando un total de 72 patas. Su memoria interna es de 256 kB, igual a la de la Arduino MKR1000. Tiene un precio oficial de unos 45 €, pero se pueden encontrar copias desde 10 €. Su principal desventaja es que no dispone de conexión a internet. Para ello, se tiene que utilizar conjuntamente un módulo que se la proporcione. El dispositivo más utilizado para este fin es el ESP8266. Se trata de un módulo sencillo y barato (tiene un precio que arranca en unos 2 €) que otorga conexión mediante Wi-Fi a placas que no dispongan de ello.

La última opción es la ESP32 *development board* (9). Se trata de una placa que ha salido recientemente al mercado. No ha sido diseñada para trabajar con el entorno de Arduino,

pero aun así, se ha desarrollado un firmware que permite su programación con este entorno. Debido a ser una placa relativamente nueva y que es la comunidad de desarrolladores la que se encarga de adaptar su código, todavía existen algunos errores e incluso faltan ciertas funcionalidades.

Esta placa incorpora conexión Wi-Fi y tiene una memoria interna de 520 kB. Contiene dos procesadores, uno para las operaciones convencionales y otro dedicado exclusivamente a controlar la conexión Wi-Fi. Su procesador es diez veces más rápido que el de la Arduino Mega y cinco veces más veloz que el de la Arduino MKR1000. Tiene un total de 25 patas de entradas digitales, de las cuales 15 pueden ser también para entradas analógicas. Su precio es de unos 15 €, pero se pueden encontrar placas originales por menos de la mitad de precio.

Una vez analizadas las placas por separado, se ha procedido a descartar la MKR1000. Comparándola con la ESP32, pese a que la primera es totalmente funcional trabajando con el entorno de Arduino, tiene un menor número de patas y un precio que puede llegar a ser siete veces superior que el de la segunda placa. Además, tiene una capacidad de almacenamiento y procesamiento inferiores.

En cuanto a las dos que quedan, la Arduino Mega tiene la principal ventaja de tener algo más del doble de patas que la ESP32, aunque por otra parte su precio oficial es tres veces mayor y su precio reducido (si tenemos en cuenta el módulo ESP8266) es poco menos del doble. Además, la Arduino Mega también tiene una memoria y una velocidad de procesamiento menores. Por otra parte, la adición de un módulo externo supone un pequeño coste extra (cables, herramientas de soldadura y tiempo). Por estas razones, la placa escogida finalmente ha sido la ESP32 *development board*.

4.6. Tratamiento de los datos en internet

Una vez hecha la elección acerca de cómo enviar las mediciones a internet, hay que saber qué hacer con esos datos. En concreto, dónde y cómo almacenarlos y cómo transmitir luego esa información al usuario.

Inicialmente se pensó en crear una base de datos mediante MySQL. Las mediciones se enviarían allí y se almacenarían. Paralelamente, se pretendía desarrollar una aplicación para el sistema operativo Android haciendo uso de una plataforma llamada AppInventor. Esta plataforma permite crear una aplicación de forma sencilla y sin necesidad de programarla directamente, haciendo uso de bloques que se van uniendo y van dando forma al programa. Este sería capaz de hacer una lectura de las mediciones de la base de datos y mostrarlas

en el teléfono, ya fuera mediante un gráfico o simplemente mostrando el último valor enviado.

La principal ventaja de este sistema es el gran control que uno tiene sobre los datos, pudiendo elegir cómo almacenarlos y clasificarlos. Sin embargo, esta forma de tratamiento comporta ciertos problemas o limitaciones.

Para empezar, se necesita tener un dominio propio en internet en el que alojar la base de datos. Para ello, el cliente puede alojar sus propios servidores o bien recurrir a una empresa externa de *hosting*, que permite albergar la base de datos en sus servidores a cambio de una cierta retribución económica. También hay que tener en cuenta que este sistema limita la lectura de los datos a teléfonos con sistema operativo Android. Los móviles con otros sistemas operativos (iOS, Windows Phone…) y otros dispositivos como ordenadores (a no ser que se busque directamente en la base de datos) se verían privados de utilizar este sistema (o bien se tendrían que crear aplicaciones o programas específicos para cada plataforma).

Esta última limitación ha sido decisiva para descartar este sistema, puesto que se pretende que los datos estén disponibles para su lectura desde cualquier dispositivo de una forma sencilla. Para ello, se ha optado por enviar las mediciones a páginas dedicadas a ello.

Estas empresas ofrecen un servicio de almacenamiento y tratamiento de datos provenientes de dispositivos conectados a internet, siendo su uso muy extendido en el internet de las cosas. Por ello, y tras el auge en los últimos años de proyectos conectados a la red, este tipo de páginas ha proliferado significativamente. Debido a este aumento, existen multitud de estas empresas y en ocasiones puede ser muy complicado escoger cuál de ellas utilizar, pues su uso y funcionalidades son muy similares.

La elección de la página a utilizar debe fundamentarse en las necesidades del cliente en base al coste de la suscripción, a la capacidad de almacenamiento o al número de dispositivos permitidos, entre otros. El uso de una página u otra es muy similar y suele requerir cambiar pocas líneas de código en la placa. La elección de dicha página en este proyecto radica más en cuestiones prácticas que en factores como los mencionados anteriormente. En concreto, después de una investigación previa se ha seleccionado Adafruit IO para el envío y tratamiento de los datos.

Esta página tiene la ventaja de contar con numerosos tutoriales de aprendizaje, además de que el código para el envío de las mediciones es relativamente sencillo, por lo que resulta más fácil detectar posibles errores. Adicionalmente, dispone de elementos predeterminados para mostrar los datos (diferentes tipos de gráficos) y su visualización es compatible tanto con teléfonos móviles como con ordenadores y tabletas (siempre y cuando se disponga de

un navegador). Adafruit IO también dispone de librerías específicas para Arduino para facilitar su programación e incorpora diversos ejemplos de código en este entorno. Por todo ello, esta ha sido finalmente la página escogida para el envío de los datos y su visualización.

5. Proceso constructivo y conexionado

5.1. Conexión del sensor y el amplificador a la placa

Una vez seleccionadas todas las opciones, se ha procedido a realizar la conexión del conjunto sensor y amplificador a la placa y a realizar las primeras mediciones.

Para conectar la célula de carga con el amplificador hay que unir los cables del sensor a las patas del amplificador tal y como se muestra a continuación:

Color del cable de la célula	Entrada en el amplificador
Rojo	E+
Negro	E-
Verde	A-
Blanco	A+

Tabla 2.Conexión de los cables del sensor al amplificador

Para ello, primero se han soldado patas tanto a las entradas como a las salidas del amplificador. A continuación, se han soldado los cables del sensor a las patas de entrada. Para conectar las patas de salida con la placa, se han utilizado cables hembra-macho para luego introducirlos en la placa de prototipo. De estas salidas, Vcc se conecta a un voltaje constante de 3,3 V (cabe recordar que este es el voltaje de funcionamiento de la placa). La pata que corresponde a GND (*ground*) se conecta a tierra. Las otras dos salidas, DT (*data*) y SCK (*clock*) se conectan a patas de la placa con entrada/salida digital.

Una vez conectado el sensor, se ha procedido a realizar algunas lecturas de prueba. Como cada célula de carga es diferente y para obtener una mejor precisión, se requiere obtener un factor de escala para cada báscula en concreto. En el apartado 7.1 se muestra cómo obtenerlo, pero para estas primeras pruebas se ha utilizado uno genérico.

Al realizar estas mediciones, los resultados mostraban que cada un cierto número de lecturas aparecía un valor completamente erróneo y aparentemente aleatorio. Esto ocurría con ambos sensores, así que se descartó que fuera un problema de estos.

Se realizó una búsqueda en la red y se descubrió que el amplificador de los sensores estaba pensado para trabajar a una frecuencia máxima de 80 MHz, mientras que el procesador de la placa utilizada funciona a 160 MHz.

Para solucionar este problema, se encontró en GitHub una librería modificada del amplificador para que fuera apto para trabajar a frecuencias más altas. Después de descargar e instalar esta librería y utilizarla en el programa se solucionó el asunto.

5.2. Proceso constructivo

Para fabricar el prototipo, además de los componentes electrónicos, se ha utilizado una tabla de conglomerado de 10 mm de grosor y dos cajas de madera. También han sido necesarios cuatro tornillos M5 de 40 mm de longitud y otros cuatro de M4 y 25 mm de longitud. Se han comprado además veinticuatro tuercas, doce correspondientes a cada tipo de tornillo.

Para la construcción, en primer lugar se ha cogido la tabla y se ha cortado hasta alcanzar unas medidas de 300 x 500 mm aproximadamente. De la parte sobrante del tablón se han cortado cuatro tacos que posteriormente se han unido a la base utilizando cola blanca.

A continuación, se han realizado cuatro perforaciones de 5 mm de diámetro (dos para la sujeción de cada sensor) y dos más en cada caja (de 4 mm de diámetro). Cada conjunto de dos agujeros, de acuerdo con las especificaciones (4), están separados por una distancia de 15 mm. Las perforaciones realizadas en las cajas se han hecho de forma que su centro coincidiera con el centro de la caja. En el caso de los agujeros de la base, estos se han realizado de tal manera que, una vez se uniera la caja y la base mediante el sensor, la primera quedara centrada en la tabla. En la Figura 3 se pueden observar las perforaciones y los tacos de la base.

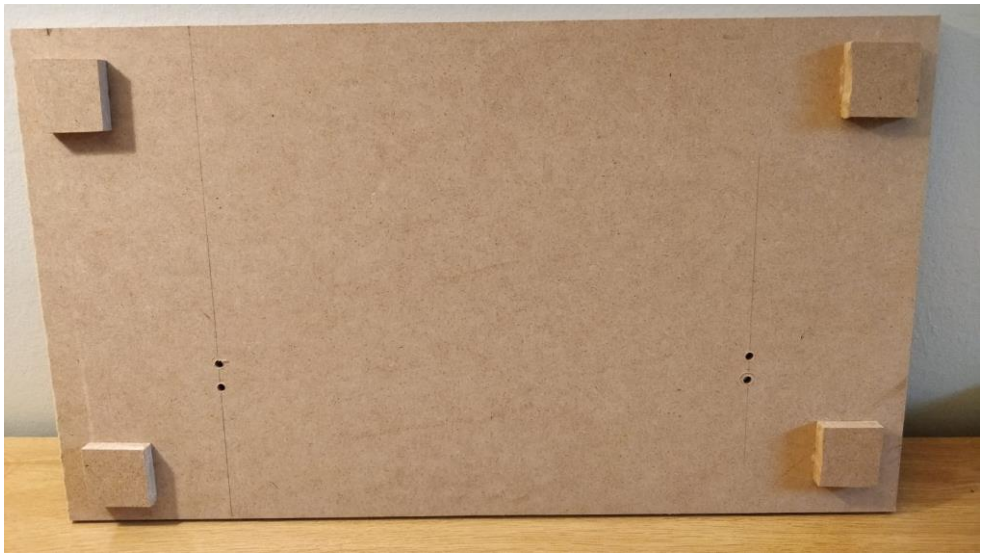

Figura 3. Perforaciones y tacos en la base

Seguidamente, los tornillos se han unido a la célula de carga y se han introducido dos tuercas para separar el sensor de la base y de la caja, de forma que pudiera flectar correctamente. Cabe destacar que los tornillos de sujeción a la base requerían ser M5, mientras que los otros debían ser M4.

El siguiente paso ha sido unir el sensor a la base y a la caja por los agujeros practicados anteriormente e insertar una tuerca al otro extremo para asegurar la sujeción. Sin embargo, después de haber instalado la caja se ha comprobado que el espacio entre esta y la base era demasiado pequeño, tal y como se puede comprobar en la Figura 5. Por ello, se ha decidido separar más el sensor de la base introduciendo una tuerca más, aumentando de esta forma la altura de la caja.

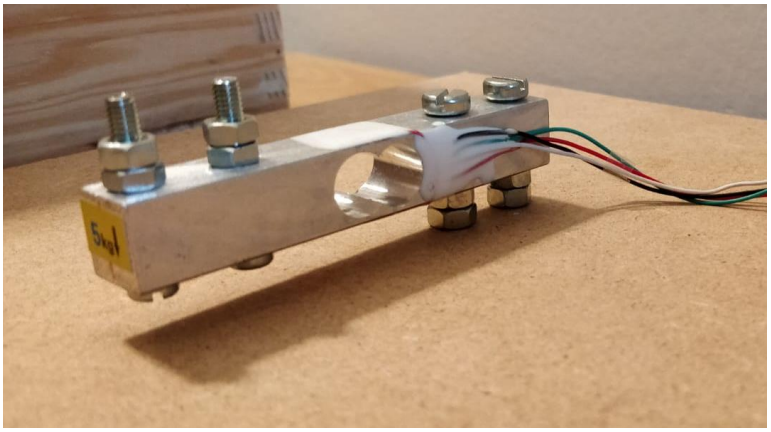

Figura 4. Sensor unido a la base mediante tornillos y tuercas

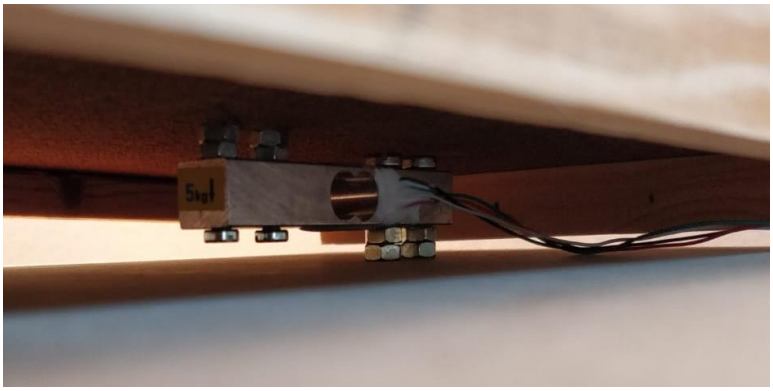

Figura 5. Caja unida a la base mediante el sensor

Finalmente, se ha unido la placa de prototipo (que incorpora la placa y el zumbador) a la base utilizando el adhesivo que lleva incorporado y se han unido los cables de los amplificadores a los lugares correspondientes en la placa de prototipo. Para ilustrar estas

conexiones[2], se ha realizado un esquema haciendo uso del programa Fritzing®.

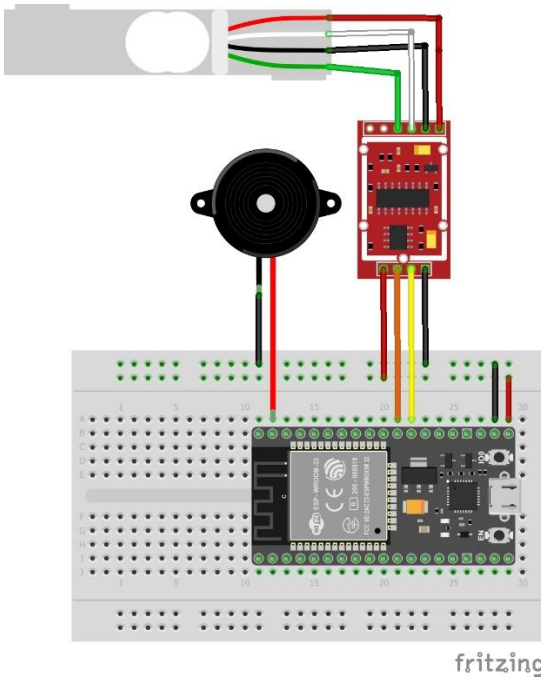

Figura 6. Conexión del zumbador y de un sensor a la placa

[2] Las patas utilizadas en el esquema no tienen por qué corresponder con las usadas en el prototipo, pues cualquier pata de entrada/salida digital sirve para conectar tanto el zumbador como los sensores.

6. Funcionalidades añadidas

Una vez finalizado el diseño básico del sistema de básculas se ha pensado en implementar nuevas funcionalidades que mejoren ciertos aspectos del proyecto. Por una parte, los esfuerzos se han concentrado en evitar que el peso soportado por las células de carga supere su valor de carga máxima. Por otra parte, se ha implementado un sistema de avisos en el teléfono móvil cuando un producto baje de cierto peso (o de ciertas unidades).

6.1. Avisador de carga máxima

Las células de carga hacen uso de galgas extensométricas para determinar la deformación sufrida y posteriormente convertir ese valor en peso soportado. Tanto esas galgas como la célula en sí tienen un rango de trabajo, en el que la tensión y la deformación son proporcionales. Sin embargo, si se supera la carga máxima (tensión de límite elástico), el metal pasa a trabajar en régimen plástico, en el que sufre deformaciones permanentes que pueden inutilizarlo como sensor. Por ello, es importante evitar que se supere ese límite.

Para ello, al principio se barajaron dos opciones: colocar un LED en cada báscula que se encendiera cuando se superara ese límite, o bien utilizar un único generador de sonidos para todas las básculas.

Las ventajas de utilizar LED radican en poder localizar fácilmente la báscula que está soportando un peso demasiado elevado en caso de estar reponiendo diferentes productos a la vez. Sin embargo, el hecho de utilizar un LED para cada báscula supone un uso de componentes mucho mayor (cables y resistencias), además de que se necesitaría ocupar una pata adicional por cada báscula.

El generador de sonidos, en cambio, no permite detectar la báscula en peligro. Sin embargo, es difícil que se esté reponiendo más de un producto a la vez, por lo que realmente esto no sería un problema. Aun así, se podrían dividir las básculas por zonas (por ejemplo, en función del producto) y colocar un generador de sonidos en cada zona. Además, este sistema tiene la ventaja de no necesitar apenas componentes (se puede colocar directamente en la placa de prototipo). Otra ventaja que tiene utilizar este componente es la facilidad para el operario, que no tiene que estar pendiente de si el LED se enciende y se puede dedicar a reponer el producto sin preocuparse por eso.

Puesto que la célula de carga utilizada en el prototipo soporta una carga máxima de 5 kg, y como el conjunto de la caja y los tornillos que la sujetan al sensor tienen un peso aproximado de 850 g, se ha escogido un peso máximo de 4 kg, dejando un margen de

seguridad de unos 150 g.

Para producir el sonido se ha utilizado la función EasyBuzzer.beep(frecuencia) de la librería EasyBuzzer. Se han realizado pruebas con diferentes frecuencias para encontrar una adecuada que no moleste al oído y finalmente 500 Hz ha sido la escogida. Por otra parte, se ha optado por un sonido intermitente. En el apartado 7.2 se explica que el bucle del programa se repite cada cuatro segundos (más el tiempo que tarda la placa en realizar las operaciones requeridas). Por ello, se ha optado por producir cuatro sonidos de 250 ms (con sus respectivos silencios de la misma duración), y esperar a continuación otros dos segundos antes de reiniciar el bucle.

6.2. Avisador de inventario mínimo

Habitualmente, en un sistema de gestión de inventario se define una cantidad de producto que marca el punto en el que se tiene que realizar un nuevo pedido. Esto se define como punto de pedido. En este proyecto se ha incorporado un sistema de avisos de punto de pedido mediante notificaciones en el móvil.

La plataforma Adafruit IO incorpora un sistema de alertas con varias opciones de configuración. Sin embargo, los avisos se realizan mediante un mensaje dentro de la propia página (lo que requeriría estar siempre conectado a esa página) o bien mediante el envío de un correo electrónico.

Esta segunda opción podría ser factible, puesto que al recibir un correo habitualmente se recibe conjuntamente una notificación. Sin embargo, eso supondría tener la bandeja de entrada llena de correos y tener que eliminarlos regularmente.

Se buscó entonces otra forma para poder enviar esos avisos y se encontró una plataforma llamada IFTTT (del inglés *if this then that*). Esta puede trabajar con los datos de Adafruit IO. Dispone además de una aplicación tanto para Android como para iOS.

Para configurar estos avisos, se tienen que crear *applets* o pequeños programas definiendo la variable a monitorizar, la condición de aviso y el mensaje a mostrar. En el caso concreto del producto 2, se desea que se avise cuando el inventario sea menor a ciertas unidades de producto. Para ello, introducimos que, si el valor de “Producto 2” es menor que una cierta cantidad, muestre el mensaje “Solo queda(n) *FeedValue* unidad(es) de producto 2”, donde *FeedValue* es el valor de la variable en ese momento. En la Figura 7 se muestra la configuración en IFTTT para el caso de menos de dos unidades.

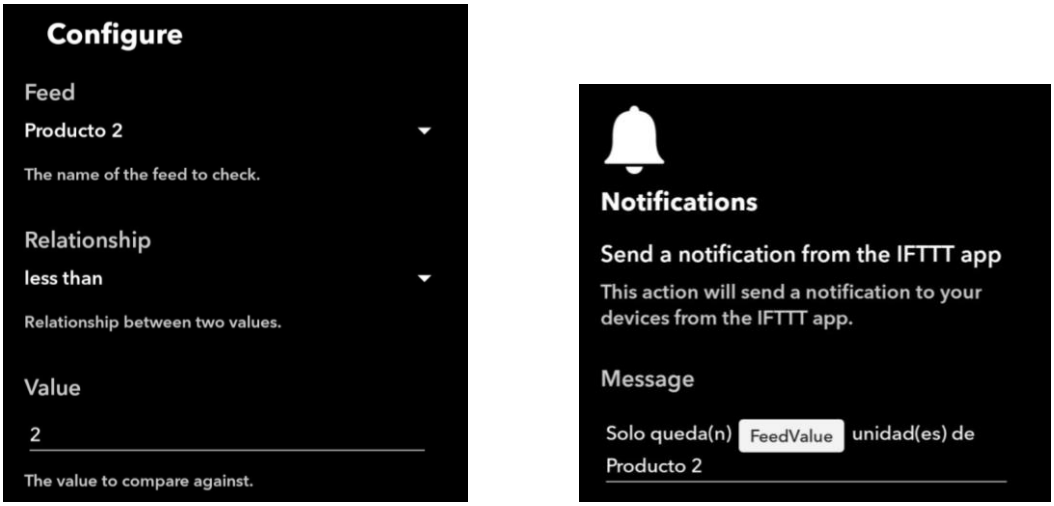

Figura 7. Configuración de un aviso en IFTTT

Este sistema tiene como principal contrapartida un tiempo de refresco de una hora, es decir, que comprueba los datos de Adafruit IO cada hora. Sin embargo, los pedidos no se suelen realizar con un margen de minutos y se ha considerado que recibir la notificación con un máximo de una hora de retraso es aceptable.

7. Programación

7.1. Programa para el cálculo del factor de escala

Para que las células de carga muestren un valor preciso se tienen que calibrar individualmente mediante un factor de escala. Para calcularlo, se ha desarrollado un programa que hace uso de la conexión serie de la placa con un ordenador. Este programa permite obtener el factor de escala de los sensores con una masa cualquiera de valor conocido. A continuación se muestran las diferentes partes de dicho programa:

```
#include "HX711.h"

HX711 scale;    //nombramos la báscula

#define DOUT 25   //nombramos las patas de datos y del reloj
#define SCK 26

void next_step(){   //función auxiliar para esperar a que el operario envíe
  while (!Serial.available()) {   //datos y luego eliminarlos del búfer
  }
  while(Serial.available()){
    Serial.read();    //mientras haya datos, los va eliminando
  }
}

void setup() {
  Serial.begin(115200);   //iniciamos la comunicación serie

  scale.begin(DOUT,SCK);    //iniciamos la conexión con la báscula

  scale.set_scale(1);    //definimos un factor de escala de 1
  Serial.print("Introduzca la salida de datos de la báscula (DT) en la");
  Serial.println(" pata 25 y la salida del reloj (SCK) en la 26");
  Serial.print("Retire todo objeto de la báscula y pulse una tecla");
  Serial.println(" cualquiera. A continuación, pulse 'enviar'");


  next_step();

  scale.tare();   //hacemos una tara
```

Figura 8. Inicio del programa

Primeramente se define el nombre de la báscula y las patas a utilizar. A continuación, se crea la función auxiliar next_step(). Su propósito general es hacer avanzar el programa cuando el operario envía un valor por el búfer. Para ello, primero la placa se espera a que el

operario mande algún valor por la conexión serie. A continuación, y mientras haya datos en el búfer, el programa los va eliminando mediante el comando Serial.read(). Esto se ha definido así para que, si el operario se equivoca y envía más de un carácter, esto no afecte al funcionamiento posterior del programa.

Comienza entonces la parte principal del programa. Se inicia la comunicación serie con el ordenador y también se inicializa la báscula con la función scale.begin(DOUT, SCK) implementada en la librería del amplificador, donde "scale" es el nombre de la báscula, mientras que DOUT y SCK corresponden a las patas de datos y del reloj, respectivamente.

El programa espera entonces a que el operario introduzca un valor cualquiera (lo que significa que ha retirado todo objeto de la báscula) para poder hacer la tara y se utiliza para ello la función auxiliar next_step(). Puesto que no se introduce ningún factor de escala en la función scale.set_scale() (donde scale es el nombre de la báscula), ya que eso es precisamente lo que se busca, el programa escoge el valor de 1. Seguidamente se realiza la tara utilizando el comando scale.tare(). Estas dos funciones están implementadas en la librería del amplificador.

```
Serial.print("\nAhora coloque un objeto de masa conocida e");
Serial.println(" introduzca su valor en g (sin decimales).");
Serial.print("Seguidamente, pulse la barra espaciadora y");
Serial.println(" haga clic en 'enviar'");

int peso = 0;   //definimos la variable del peso introducido
int c = 0;
while (c != 32) {   //cuando se pulsa la barra espaciadora se termina el bucle
  if (Serial.available()) {   //esperamos a que haya un valor para leer
    c = Serial.read();    //realizamos la lectura del carácter
    if (c > 47 and c < 58) {    //si el caracter corresponde a un número
      peso = peso*10+c-48;    //actualizamos el valor del peso
    }
  }
}

Serial.print("\nEl valor introducido es ");   //mostramos el valor del peso
Serial.println(peso);                        //para comprovar que sea correcto
Serial.print("\nA continuación se muestran las lecturas de la báscula.");
Serial.println(" El valor es únicamente orientativo");
Serial.print("Si algún valor difiere excesivamente del resto,");
Serial.println(" revise las conexiones y reinicie el programa");

while(Serial.available()){
  Serial.read();
}
```

Figura 9. Introducción del valor del peso

El programa inicia entonces un bucle que espera a que haya un valor en el búfer y lo añade a la variable peso. Este bucle finaliza cuando el operario pulsa la barra espaciadora. Cuando se introduce un carácter, Arduino lee su código ASCII. En la Figura 10 se puede observar que el número 0 corresponde al decimal 48, el 1 al 49 y así sucesivamente. Por eso, para pasar del número decimal al valor deseado se tiene que restar 48 al valor leído por la placa.

Dado que el comando Serial.read() tan solo puede leer un carácter, el programa multiplica por diez el valor anterior de la variable peso antes de sumarle el nuevo dígito (que se añade en las unidades). De esta forma se consigue generar el número del peso.

DEC	HEX	OCT	CHAR	DEC	HEX	OCT	CH	DEC	HEX	OCT	CH	DEC	HEX	OCT	CH
0	0	000	NUL	32	20	040		64	40	100	@	96	60	140	`
1	1	001	SOH	33	21	041	!	65	41	101	A	97	61	141	a
2	2	002	STX	34	22	042	"	66	42	102	B	98	62	142	b
3	3	003	ETX	35	23	043	#	67	43	103	C	99	63	143	c
4	4	004	EOT	36	24	044	$	68	44	104	D	100	64	144	d
5	5	005	ENQ	37	25	045	%	69	45	105	E	101	65	145	e
6	6	006	ACK	38	26	046	&	70	46	106	F	102	66	146	f
7	7	007	BEL	39	27	047	'	71	47	107	G	103	67	147	g
8	8	010	BS	40	28	050	(	72	48	110	H	104	68	150	h
9	9	011	TAB	41	29	051	)	73	49	111	I	105	69	151	i
10	A	012	LF	42	2A	052	*	74	4A	112	J	106	6A	152	j
11	B	013	VT	43	2B	053	+	75	4B	113	K	107	6B	153	k
12	C	014	FF	44	2C	054	,	76	4C	114	L	108	6C	154	l
13	D	015	CR	45	2D	055	-	77	4D	115	M	109	6D	155	m
14	E	016	SO	46	2E	056	.	78	4E	116	N	110	6E	156	n
15	F	017	SI	47	2F	057	/	79	4F	117	O	111	6F	157	o
16	10	020	DLE	48	30	060	0	80	50	120	80	112	70	160	p
17	11	021	DC1	49	31	061	1	81	51	121	Q	113	71	161	q
18	12	022	DC2	50	32	062	2	82	52	122	R	114	72	162	r
19	13	023	DC3	51	33	063	3	83	53	123	S	115	73	163	s
20	14	024	DC4	52	34	064	4	84	54	124	T	116	74	164	t
21	15	025	NAK	53	35	065	5	85	55	125	U	117	75	165	u
22	16	026	SYN	54	36	066	6	86	56	126	V	118	76	166	v
23	17	027	ETB	55	37	067	7	87	57	127	W	119	77	167	w
24	18	030	CAN	56	38	070	8	88	58	130	X	120	78	170	x
25	19	031	EM)	57	39	071	9	89	59	131	Y	121	79	171	y
26	1A	032	SUB	58	3A	072	:	90	5A	132	Z	122	7A	172	z
27	1B	033	ESC	59	3B	073	;	91	5B	133	[	123	7B	173	{
28	1C	034	FS	60	3C	074	<	92	5C	134	\	124	7C	174	|
29	1D	035	GS	61	3D	075	=	93	5D	135	]	125	7D	175	}
30	1E	036	RS	62	3E	076	>	94	5E	136	^	126	7E	176	~
31	1F	037	US	63	3F	077	?	95	5F	137	_	127	7F	177	DEL

Figura 10. Tabla ASCII[3]

Por último, se muestra por el monitor serie el valor introducido para comprobar que este sea correcto.

[3] http://www.webtutoriales.com/articulos/mapa-de-caracteres-y-tabla-ascii

```
int i=1;
float suma_lecturas = 0;   //variable para acumular el valor de las lecturas

while (i<= 4) {   //realizaremos 4 mediciones para hacer la media
  float lectura=scale.get_units(20);    //realizamos la lectura
  suma_lecturas += lectura;   //actualizamos la variable
  Serial.print("\nLectura: ");
  Serial.println(lectura,1);    //mostramos la lectura con 1 decimal
  if (i< 4) {
    Serial.print("Retire el objeto, vuelva a colocarlo y pulse una tecla");
    Serial.println(" cualquiera. Seguidamente, pulse 'enviar'");
    next_step();
  }
  i++;    //actualizamos el contador
}
```

Figura 11. Realización de varias mediciones.

En la siguiente parte del programa se define la variable suma_lecturas para acumular el valor de las mediciones. A continuación, se inicia un bucle de cuatro iteraciones en que se realizan las lecturas mediante el comando scale.get_units(X), donde scale es el nombre de la báscula y X es el número de mediciones consecutivas que realiza el programa, devolviendo el valor medio de estas. Esta función está implementada en la librería del amplificador.

Para obtener un factor de escala más preciso, se requiere que se retire y se coloque la masa en cuatro ocasiones. Con este propósito, se pide al operario que envíe un carácter cualquiera cuando lo haya hecho para que el programa continúe con la siguiente lectura, utilizando para esto la función next_step() previamente creada. Después de cada medición, esta se muestra por la pantalla para comprobar que no difiera demasiado de las demás, lo que podría indicar un posible error. Este valor es únicamente orientativo.

```
  float factor_escala = suma_lecturas/(peso*(i-1));
  Serial.print("\nEl factor de escala es ");   //calculamos y mostramos el
  Serial.println(factor_escala,3);   //factor de escala con tres decimales
  }

void loop() {
}
```

Figura 12. Obtención del factor de escala.

Finalmente, el programa realiza la media de las diferentes mediciones y muestra el factor de escala por la pantalla con tres decimales.

7.2. Programa principal

Una vez hallado el factor de calibración para cada una de las básculas, se procede a cambiar el programa de la placa para que pase a realizar lecturas y enviar esos datos a la nube.

Para este prototipo se ha propuesto un programa en el que únicamente participan dos básculas, pero se puede extender de forma análoga para más sensores. A continuación se muestra dicho programa.

Programa_final_2 | config.h §

```
#define IO_USERNAME    "nombre_usuario"
#define IO_KEY         "clave_adafruit_io"

#define WIFI_SSID       "SSID"
#define WIFI_PASS       "contraseña"

#include "AdafruitIO_WiFi.h"    //https://github.com/adafruit/Adafruit_IO_Arduino
AdafruitIO_WiFi io(IO_USERNAME, IO_KEY, WIFI_SSID, WIFI_PASS);
```

Figura 13. Configuración de la conexión Wi-Fi y de la cuenta de Adafruit IO

En la Figura 13 se puede ver una parte del programa en la que se definen los parámetros de la conexión Wi-Fi así como el usuario y la clave de Adafrit IO. Después se incluye la librería AdafruitIO_Wifi (que viene dentro de Adafruit IO Arduino[4]) y se crea una instancia de un objeto del tipo AdafruitIO_WiFi llamada io() y que incluye los parámetros anteriormente mencionados. Esto configura la conexión con Adafruit IO (10).

Además de AdafruitIO_Wifi, en el programa también se ha utilizado la librería del amplificador HX711[5] modificada para que pueda trabajar con esta placa y EasyBuzzer[6], utilizada debido a que la función tone () (usada para producir los sonidos) no se encuentra disponible todavía para la placa utilizada en este proyecto. Por ello, se ha tenido que buscar otra forma de generarlos, y finalmente esta librería ha sido la opción escogida, ya que está adaptada para poder funcionar en esta placa.

En la siguiente página se muestra la parte principal del programa.

[4] Adafruit IO Arduino: https://github.com/adafruit/Adafruit_IO_Arduino

[5] HX711, modificada por Geert Roumen (Lemio): https://github.com/lemio/HX711b

[6] EasyBuzzer, creada por Evert Arias: https://github.com/evert-arias/EasyBuzzer

```
#include "config.h"
#include "HX711.h"    //https://github.com/lemio/HX711b
#include "EasyBuzzer.h"   //https://github.com/evert-arias/EasyBuzzer

HX711 scale1;
HX711 scale2;   //definimos los nombres de las dos básculas

#define DOUT1 25
#define SCK1 26

#define DOUT2 32
#define SCK2 33   // definimos las patas de las básculas

#define TPIN 27   // definimos la pata del generador de sonidos

#define freq 500

AdafruitIO_Feed *producto1 = io.feed("Producto 1");
AdafruitIO_Feed *producto2 = io.feed("Producto 2");

void bip () {
  EasyBuzzer.beep(freq);    //producimos un sonido de 500 Hz
  delay(250);
  EasyBuzzer.stopBeep();
  delay (250);
}

void setup() {

  scale1.begin(DOUT1, SCK1);
  scale2.begin(DOUT2, SCK2);    //inicializamos las básculas

  EasyBuzzer.setPin(TPIN);    //configuramos la pata para el zumbador

  io.connect();   //iniciamos la conexión con Adafruit IO
  while(io.status() < AIO_CONNECTED) {
    delay(500);   //esperamos hasta que se conecte
  }
  //estamos conectados

  scale1.set_scale(-396.403);
  scale2.set_scale(-378.132);    //introducimos el factor de escala
  scale1.tare();
  scale2.tare();    //hacemos una tara

}
```

Figura 14. Definición de variables e inicialización del programa

Aquí se incluye la parte de configuración config.h, se definen los nombres de las básculas, se asigna un nombre a las patas en uso y se define la relación entre las variables en el programa y en Adafruit IO mediante la función implementada en su librería AdafruitIO_Feed *producto1=io.feed(Producto 1). En esta función, “producto1” hace referencia a la variable

dentro del programa, mientras que “Producto 1” se refiere a la de Adafruit IO. Seguidamente se crea la función bip(), que genera un sonido de una frecuencia concreta y de 250 ms de duración, y luego espera otros 250 ms.

A continuación, se inicializan las básculas con la función scale1.begin (DOUT1, SCK1) explicada en el apartado anterior. También se define la pata del zumbador con EasyBuzzer.setPin(número_pata), se inicia la conexión con Adafruit IO mediante io.connect() (implementada en la librería), se introduce el factor de escala utilizando nombre_báscula.set_scale(factor_escala) y se taran las básculas con nombre_báscula.tare(). Estas dos últimas funciones provienen de la librería del amplificador.

```
void loop() {

  io.run();
  EasyBuzzer.update();

  scale1.power_up();    //encendemos el amplificador 1
  int weight1 = scale1.get_units(20);    //realizamos 20 lecturas
  producto1->save(weight1);   //enviamos los datos del producto 1
  scale1.power_down();    //apagamos el amplificador para ahorrar energía

  scale2.power_up();    //encendemos el amplificador 2
  int weight2 = scale2.get_units(20);    //realizamos 20 lecturas
  producto2->save(weight2);   //enviamos los datos del producto 2
  scale2.power_down();    //apagamos el amplificador para ahorrar energía
  if (weight1>4000 or weight2>4000) {   //si alguna báscula tiene sobrepeso
    for (int i = 1; i <= 4; i++){       //producimos cuatro pitidos
      bip ();
    }
  }
  else {
    delay (2000);
  }
  delay (2000);
}
```

Figura 15. Realización de las lecturas y envío de los datos.

En esta segunda parte del programa primero se incluyen dos funciones necesarias para el correcto funcionamiento de ciertas librerías dentro del programa. En concreto, Adafruit IO requiere io.run() mientras que EasyBuzzer necesita la función EasyBuzzer.update().

A continuación, en cada báscula se enciende el amplificador, se realiza la medición usando scale.get_units(20) (esta función está explicada en el apartado anterior) y se envía mediante el comando variable->save(valor) (implementado en la librería de Adafruit IO). Seguidamente se apaga el amplificador y, si el peso de alguna báscula supera un valor

límite, se llama cuatro veces la función bip(). Si, por el contrario, no existe sobrepeso, el programa deja pasar dos segundos. Finalmente se dejan pasar otros dos segundos más antes de empezar el bucle de nuevo, ya que no es necesario actualizar constantemente el valor del peso.

En el caso de almacenar unidades de producto con un peso determinado (por ejemplo, paquetes de pasta de 500 g), se debería realizar una pequeña modificación en el programa tal y como se muestra en la Figura 16.

```
scale2.power_up();    //encendemos el amplificador 2
int weight2 = scale2.get_units(20);    // hacemos la media de 20 lecturas
int n2=round(weight2/500);   //calculamos el número de paquetes de 500 g que tenemos
producto2->save(n2);   //enviamos los datos del producto 2
scale2.power_down();    //apagamos el amplificador para ahorrar energía
```

Figura 16. Modificación del programa para mostrar unidades de un producto

En este caso, el producto se encuentra en la báscula 2. Se realiza la medición del peso y luego se divide entre el peso total del producto (incluyendo el peso del envase). En el caso concreto de este ejemplo, debido a que el envase es un envoltorio de plástico, apenas añade peso al total y por eso se ha utilizado el valor nominal de 500 g.

Puesto que las lecturas pueden no ser del todo exactas o bien el peso del producto puede ser ligeramente diferente del valor introducido, el resultado de la división se redondea con el comando round(peso_total/peso_nominal). Finalmente, este resultado se envía a Adafruit IO.

7.3. Configuración de Adafruit IO

7.3.1. Configuración de las variables

Llegados a este punto, el siguiente paso es la configuración de la página Adafruit IO. Para ello, primero se necesita crear una cuenta. Cada usuario dispone de una clave asociada a esa cuenta que se tiene que introducir en el apartado config.h del programa principal. Para obtenerla, solo hay que pulsar “View AIO Key” en la página principal de Adafruit.

El siguiente paso es crear una variable (*feed* en Adafruit IO) (11). Para ello, hay que pulsar “*Feeds* → *Actions* → *Create a new feed*” tal y como se ve en la Figura 17. Seguidamente hay que nombrarlo y, opcionalmente, añadir una descripción. En este proyecto, las variables son Producto 1 y Producto 2.

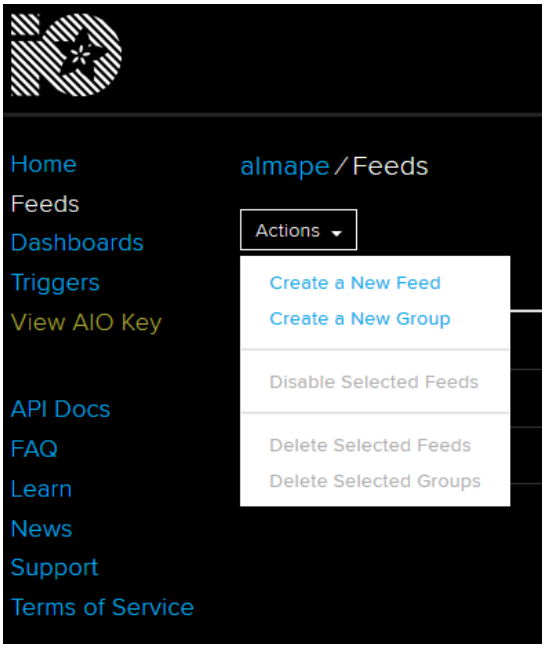

Figura 17. Creación de un nuevo feed

Una vez creada la variable, se procede a su configuración pulsando sobre ella. A continuación se muestran los diferentes parámetros que se pueden modificar.

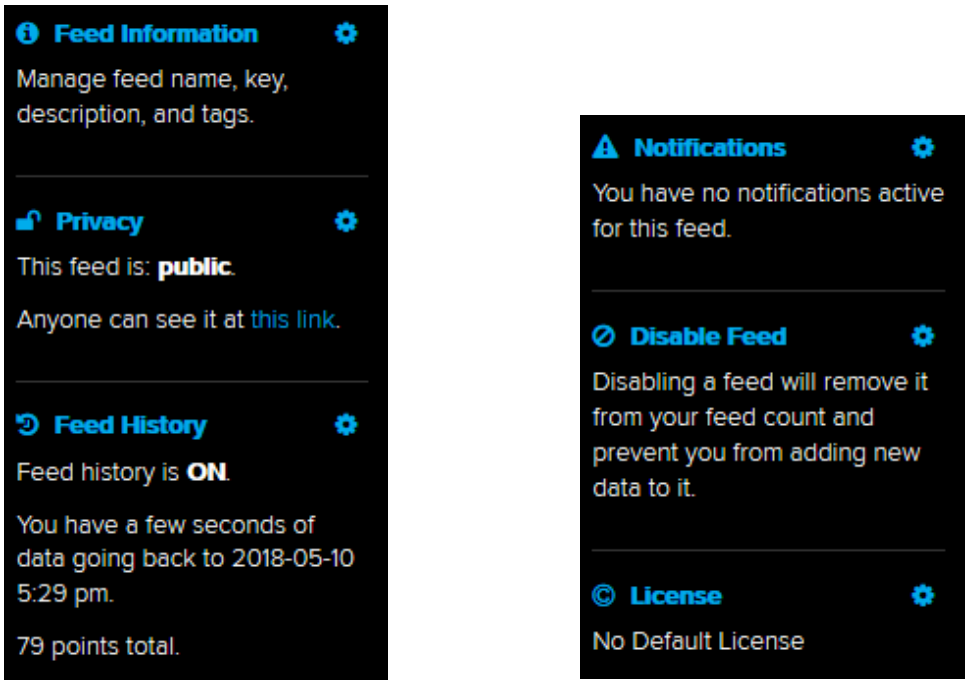

Figura 18. Opciones de configuración de la variable

En la parte de *feed information* se puede cambiar el nombre, la descripción y se pueden añadir etiquetas. En *privacy* se puede seleccionar si los datos son privados o públicos. En este proyecto se ha escogido la segunda opción para que varios usuarios puedan acceder a las lecturas de los sensores. En la parte de *feed history* se selecciona si se quiere un registro de las mediciones pasadas o solo se precisa el valor actual. Puesto que al cliente podría interesarle saber el comportamiento del inventario, se ha habilitado esta opción.

Otra opción es el envío de un correo si se reciben nuevas mediciones, pero esta resulta poco útil en este proyecto, puesto que las básculas envían datos constantemente. También existe la posibilidad de desactivar la variable (evitando así que reciba nuevos datos) o de otorgarle una licencia.

Por último, cabe la posibilidad de descargarse todos los datos recibidos por esa variable. Para ello, hay que pulsar "*Actions* → *Download all data"* y, a continuación, hay que escoger el tipo de documento en el que incluir la información (CSV o JSON).

7.3.2. Configuración de los *dashboards*

Para este apartado se van a utilizar dos términos ingleses que no tienen traducción directa al castellano. El primero de ellos, *widget*, hace referencia a un icono o figura dinámicos (es decir, que pueden cambiar con el tiempo) y con el que, en ocasiones, se puede interactuar (como es el caso de los *widgets* en los teléfonos inteligentes). El segundo elemento es el *dashboard*, que hace referencia al panel en el que se encuentran estos *widgets*.

Una vez configuradas las variables hay que hacer lo propio con los *dashboards* (12). Para ello, primero hay que pulsar "*Dashboards* → *Actions* → *Create a new dashboard*". A continuación, se le asigna un nombre y opcionalmente se añade una descripción. En este proyecto se ha decidido crear tres *dashboards*: uno en el que se muestre información de todos los productos a la vez y otros dos más para cada producto en concreto. Cabe destacar que el número y la configuración de los *dashboards* debe responder a los requerimientos del cliente y estos pueden ser modificados según sus preferencias.

Una vez creados los *dashboards* se añaden los *widgets*. De los botones que se muestran en la Figura 19, situados en la parte superior derecha de la pantalla, hay que pulsar en el de color azul con el símbolo +.

Figura 19. Botones para la configuración del dashboard

A continuación, se abre una ventana con los diferentes *widgets* disponibles (Figura 20). En este proyecto solo se van a utilizar el calibre (*gauge*) y el gráfico (*line chart*). En concreto, en el *dashboard* de todos los productos únicamente se colocarán calibres (puesto que se pretende dar una visión general del inventario en ese momento), mientras que en el *dashboard* del producto en concreto se colocará, además del calibre, un gráfico con las diferentes mediciones realizadas a lo largo de un tiempo determinado. Una vez seleccionado el *widget* hay que configurarlo con las unidades, los valores máximo y mínimo y, en el caso del gráfico, el tiempo que se pretende mostrar.

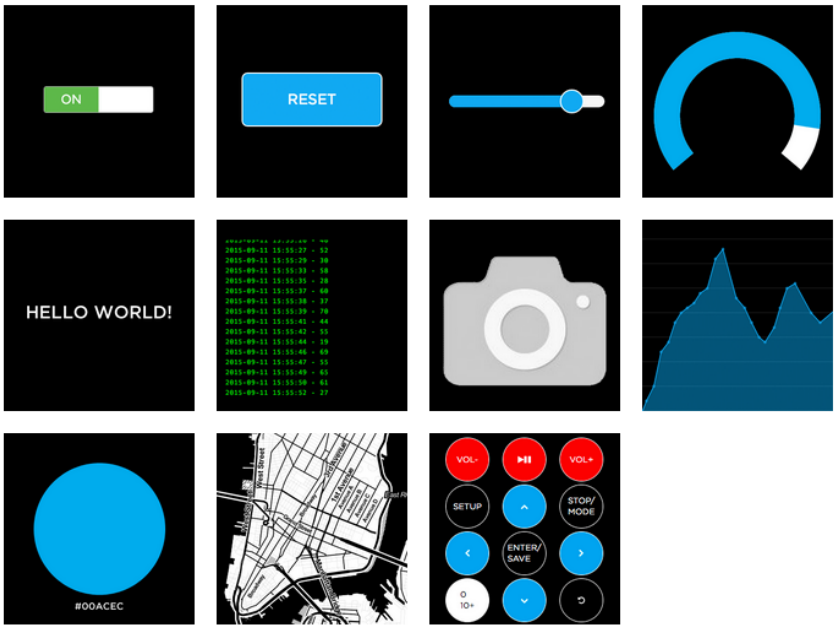

Figura 20. Widgets disponibles en Adafruit IO

Para editar el *dashboard* hay que pulsar el botón verde que aparece en la Figura 19. Con esto se puede cambiar la configuración de los *widgets*, su posición y su tamaño. Pulsando el botón rojo se consigue borrar el *dashboard*, mientras que el amarillo muestra la clave personal de Adafruit IO. El quinto botón muestra el enlace del *dashboard* en concreto, mediante el cual se puede acceder a este. Si el *dashboard* es público, lo que se consigue pulsando el sexto botón, cualquier persona que disponga del enlace puede acceder al *dashboard*. Finalmente, el último botón sirve para mostrar los *widgets* en pantalla completa.

Una de las virtudes de Adafruit IO es que adapta estos *dashboards* a un formato adecuado cuando se accede a él mediante un teléfono móvil. En la Figura 21 se muestra el *dashboard* del producto 2 en este formato.

Figura 21. Dashboard del producto 2 visto des de un teléfono móvil

8. Evaluación de costes

Para realizar una valoración general del presupuesto de este proyecto, se ha tenido en cuenta tanto el coste material como el coste laboral.

Componente	Coste por unidad (€/unidad)	Unidades	Coste total (€)
Placa	5,98	1	5,98
Célula de carga y amplificador	3,27	2	6,54
Zumbador	1,72	1	1,72
Placa de prototipo	1,99	1	1,99
Base de madera	4,54	1	4,54
Caja de madera	7,99	2	15,98
Tornillos y tuercas	-	-	2,90
Otros componentes[7]	-	-	1,00
Coste total			40,65

Tabla 3. Coste de los componentes del proyecto

Concepto	Horas	Coste (€)
Búsqueda de información previa	150	1500,00
Diseño y programación	90	900,00
Montaje	10	100,00
Coste total		2500,00

Tabla 4. Coste laboral del proyecto

[7] Cables, estaño para soldar, patas, tacos de madera, etc.

Para calcular el coste laboral de este proyecto, se ha supuesto un precio de 10 €/h. Cabe destacar que en la estimación no se han tenido en cuenta las horas de trabajo para la redacción de la memoria.

Tipo de coste	Coste (€)
Material	40,65
Laboral	2500,00
Coste total del proyecto	2540,65

Tabla 5. Coste total del proyecto

Finalmente se ha calculado el coste total del proyecto. A este valor se tendría que añadir, en función de los requerimientos del cliente, el coste de suscripción a la página de envío de datos. Pero como en este prototipo se ha optado por una cuenta básica sin coste, no se ha incluido esta partida. Otro factor a tener en cuenta sería el coste del *software*, pero todos los programas y librerías utilizados son de uso gratuito.

Conclusiones

El principal objetivo de este proyecto ha sido crear un prototipo de un sistema de control de inventario conectado a la red. Para ello, se han utilizado dos células de carga conectadas a sus respectivos amplificadores y luego a la placa. Una vez realizadas las lecturas, los datos de los sensores se envían a la plataforma Adafruit IO, que los almacena y los puede representar utilizando figuras y gráficos. Además, esta plataforma está adaptada para su uso en dispositivos móviles, así que el objetivo de poder acceder a las mediciones desde cualquier parte también se ha cumplido.

Para realizar el proyecto se pretendía programar un microcontrolador en el entorno de Arduino, pero la gran cantidad de placas disponibles hizo que la elección de esta fuera complicada. Otro punto a tener en cuenta fue el de los sensores, pues inicialmente se pensó en usar sensores de presión, pero las dificultades que estos planteaban los hicieron inviables. Las células de carga, además de ser más complejas tanto en su conexión (necesitan un amplificador) como en su programación, presentaron ciertos problemas a la hora de realizar las lecturas, pues la placa trabajaba a una frecuencia superior a la soportada por el amplificador. Afortunadamente, se encontró una librería modificada de este amplificador que hacía posible su compatibilidad con la placa.

Otros problemas han derivado del hecho que la placa utilizada no está diseñada para trabajar con Arduino, sino que se le ha añadido un *firmware* que hace posible esa compatibilidad. Uno de estos problemas es la inexistencia de la función *tone* para generar sonidos, por lo que se ha tenido que utilizar una librería para poder producir las alertas de carga máxima.

El prototipo fabricado es plenamente funcional y por tanto el objetivo principal se ha llevado a cabo de forma exitosa. En cuanto a los objetivos personales, la programación realizada en C++ (un lenguaje previamente desconocido por el autor) y la utilización de una plataforma dedicada al internet de las cosas hacen que estos objetivos también hayan sido cumplidos.

Propuestas de mejora

Puesto que este proyecto se ha centrado en el diseño y la fabricación de un prototipo, hay ciertos aspectos de este sistema que, en una versión definitiva, se podrían mejorar.

En primer lugar, cabe destacar que una de las principales desventajas de utilizar células de carga como las de este proyecto es la gran cantidad de espacio que se necesita para su instalación, lo que reduce considerablemente la capacidad disponible para el almacenamiento. Puesto que en un almacén este es el factor principal, una posible mejora pasaría por disminuir el espacio ocupado por los sensores. Para ello, se podría reducir el volumen de las células de carga utilizando otro tipo de sensores con menor altura, o bien utilizando básculas ya integradas en la estantería.

En este último caso, un punto a tener en cuenta es la integración en la estantería no solo de la célula de carga, sino también del amplificador. Con ello, se podría disminuir ese espacio utilizado aún más, o bien redistribuirlo de forma más eficiente.

La segunda propuesta deriva de la necesidad de utilizar un ordenador portátil cada vez que se necesita realizar una recalibración de las células de carga, lo que ralentiza considerablemente este proceso. Además, el hecho de tener que volver a cargar los programas cuando se calibran las básculas, unido a la necesidad de introducir los nuevos factores manualmente, hace que esta operación sea todavía más lenta.

Una posible mejora para esto sería integrar ambos programas en uno solo. Por medio de un interruptor, se podría escoger qué parte utilizar en cada momento. Adicionalmente, se debería incorporar una pequeña pantalla y un teclado numérico básico a la placa para poder llevar a cabo la calibración. A su vez, se debería poder seleccionar la báscula a calibrar sin necesidad de cambiar las patas cada vez que se realiza esta operación. Utilizando un peso patrón se evitaría la necesidad de tener que introducirlo cada vez que se calibrara una báscula. Finalmente, el programa calcularía automáticamente el factor de escala y el programa principal podría disponer de él sin que un operario tuviera que introducirlo manualmente.

En tercer lugar, en el caso de tener una gran cantidad de básculas (como en almacenes grandes), convendría utilizar un concentrador para recibir los datos de todas las básculas. Esta opción es especialmente interesante si se integran los sensores y los amplificadores en la estantería, pues de ella podría salir un único cable con la información de todas ellas. A su vez, los distintos cables irían a un ordenador central, desde donde se enviarían los datos a la red.

De esta forma se conseguiría reducir aún más el espacio ocupado por el sistema, ya que no sería necesario instalar múltiples placas ni emisores Wi-Fi para emitir las señales.

Agradecimientos

En primer lugar, querría agradecer a Oriol Boix, tutor de este trabajo, su dedicación en el seguimiento y en la corrección de la redacción de este proyecto. También agradecerle su ayuda en ámbitos de la informática y las telecomunicaciones con los que yo apenas estaba familiarizado, así como sus consejos a la hora de tomar decisiones importantes.

También querría agradecer la ayuda de Hermes Valenciano en los compases iniciales del proyecto en temas como la elección de la placa o mis dudas sobre programación.

Por último, agradecer a Jaume Carrascosa su consejo en algunos aspectos de la programación en C++ y en la mejora de la eficiencia de los programas.

Bibliografía

Referencias bibliográficas

1. **Arduino.** *Arduino UNO Rev3.* [En línea] [Consultado el: 21 de febrero de 2018.] https://store.arduino.cc/arduino-uno-rev3.

2. **RaspBerry Pi.** *Raspberry Pi 3 Model B.* [En línea] [Consultado el: 21 de febrero de 2018.] https://www.raspberrypi.org/products/raspberry-pi-3-model-b/.

3. **Interlink electronics.** Sparkfun. *Force sensing resistor. Integration guide and evaluation parts catalogue.* [En línea] [Consultado el: 4 de marzo de 2018.] https://www.sparkfun.com/datasheets/Sensors/Pressure/fsrguide.pdf.

4. **HTC Sensors.** Sparkfun. *Parallel beam load cell TAL220.* [En línea] [Consultado el: 4 de marzo de 2018.] https://cdn.sparkfun.com/datasheets/Sensors/ForceFlex/TAL220M4M5Update.pdf.

5. **Martín, Eva.** Tuexperto. *Qué es Bluetooth y para qué sirve.* [En línea] 2013. [Consultado el: 2018 de febrero de 28.] https://www.tuexperto.com/2013/05/06/que-es-bluetooth-y-para-que-sirve/.

6. **Test de velocidad.** *WiFi vs cable de red ethernet. ¿Cuál es la mejor opción?* [En línea] 2017. [Consultado el: 4 de marzo de 2018.] https://www.testdevelocidad.es/2017/06/06/wifi-vs-cable/.

7. **Arduino.** *Arduino MKR1000 WIFI.* [En línea] [Consultado el: 12 de marzo de 2018.] https://store.arduino.cc/arduino-mkr1000.

8. **Arduino.** *Arduino Mega 2560 Rev3.* [En línea] [Consultado el: 12 de marzo de 2018.] https://store.arduino.cc/arduino-mega-2560-rev3.

9. **Vaduva Ionut Lucian.** *ESP32 – Cheapest IoT WiFi and Bluetooth ready module.* [En línea] 2018. [Consultado el: 14 de marzo de 2018.] https://www.geekstips.com/esp32-review-idf-programming-tutorial/.

10. **Adafruit Industries.** *Adafruit IO Basics: Analog Input.* [En línea] 2017. [Consultado el: 3 de abril de 2018.] https://learn.adafruit.com/adafruit-io-basics-analog-input.

11. **Adafruit Industries.** *Adafruit IO Basics: Feeds.* [En línea] 2017. [Consultado el: 10 de abril de 2018.] https://learn.adafruit.com/adafruit-io-basics-feeds.

12. **Adafruit Industries.** *Adafruit IO Basics: Dashboards.* [En línea] 2017. [Consultado el: 10 de abril de 2018.] https://learn.adafruit.com/adafruit-io-basics-dashboards.

Bibliografía complementaria

AddOhms. *Comparing the Arduino Uno and Raspberry Pi.* [En línea] 2013. [Consultado el: 21 de febrero de 2018.] https://www.youtube.com/watch?v=7vhvnaWUZjE.

Geert Roumen (Lemio). Github. *ESP32 issue, clock speed too fast for HX711.* [En línea] 2017. [Consultado el: 24 de marzo de 2018.] https://github.com/bogde/HX711/issues/75.